**TP Architecture des ordinateurs**

**2 TP1 : le corrigé**

**Observer**

A quels codes machines correspond l'instruction mov bx,0002 ? ***BB 02 00***
A quelle adresse est stockée cette instruction ? ***à l'adresse CS:0103*** ***Expliquer l'instruction précédente occupe 3 octets à partir de l'adresse CS:0100***
A quels codes machines correspond l'instruction add ax,bx ? ***03  C3***
Que peut-on dire du nombre d'octets qui codent une intruction ? ***il est variable***

Les 0 de poids forts ne sont pas significatifs lorsqu'on entre une instruction sous forme mnémonique : mov ax,1 est identique à mov ax,0001

Que peut-on dire de l'ordre de stockage des octets constituant la donnée 0001 ? ***l'octet de poids faible est stocké avant l'octet de poids fort (ordre dit "little-endian"), l'ordre de stockage est inversé***

Cet ordre de stockage est une convention du fabricant de microprocesseur (ici Intel) Certains fabricants ont adopté la convention inverse.

**Le panneau de données**

Dans le panneau de données, on peut observer la mémoire qui est visualisée par octets à la fois en hexa et en ASCII.

* Se déplacer dans le panneau de données (en bas à gauche) à l'aide de la touche TAB,
* avec Ctrl+G (G comme Goto) entrer l'adresse cs:0100 (ou cs:100 peu importe)
* que voit-on dans la mémoire ? ***On voit les codes machine des instructions entrées*** Expliquer. ***Le panneau de données ne permet pas seulement d'examiner le segment de données mais n'importe quel segment***
* Avec Ctrl+G entrer l'adresse cs:0101,
* à quelle adresse l'octet 01 est-il stocké ? ***il est stocké à l'adresse CS:0101***

**Exécuter les instructions**

Nous allons exécuter ces trois instructions pas à pas (mov ax,1   mov bx,2 et add ax,bx) et observer leur effet sur les registres IP, AX et BX du microprocesseur. Si on veut répéter plusieurs fois l'exécution dans les mêmes conditions, on peut se reporter à la rubrique suivante *Modifier les registres*.

* Avec TAB, aller dans le panneau de code,
* noter la valeur actuelle des registres AX, ***0000*** BX ***0000*** et IP ***0100***

Dans le panneau de code, un petit triangle sépare la zone des adresses et la zone des codes machine,

* noter la position du triangle, ***face à CS:0100***
* appuyer sur la touche F8, (exécute un pas de programme)
* noter la valeur du registre AX ***0001*** (panneau des registres à droite) expliquer ***on visualise l'effet de l'exécution de l'instruction mov ax,1 sur le registre ax***
* Noter la valeur du registre IP, ***0103***
* comment a évolué la position du triangle ? ***il est face à CS:0103***
* que repère-t-il ? ***le point où l'exécution est arrivée***
* Appuyer à nouveau sur F8,
* noter la valeur de BX, ***0002***
* noter la valeur de IP, ***0106***
* appuyer à nouveau sur F8,
* noter la valeur de AX ***0003***, et BX ***0002***
* noter la valeur de IP ***0108***.
* A quoi sert IP ? ***à repérer dans le segment de code, le code de la prochaine instruction à exécuter.***
* Que fait l'instruction mov ? ***elle réalise une affectation de l'opérande de droite dans l'opérande de gauche***
* Que fait l'instruction add ? ***elle réalise la somme de l'opérande de gauche et de l'opérande de droite***
* Dans quel registre le résultat est-il stocké ?  ***elle stocke le résultat dans l'opérande de gauche***

**Mise au point d'un programme**

**Observer**

* Que voit-on maintenant dans le panneau de code ? ***on voit le programme tp1.exe***
* Que voit-on notamment sous chaque ligne de C ? ***on voit les instructions assembleur correspondant au C***
* Noter l'instruction assembleur qui appelle un sous-programme (\_clrscr par exemple est un sous-programme) c'est l'instruction ***call \_clrscr***
* Noter la valeur du registre IP, et la position du triangle dans le panneau de code. ***02EA***

**Exécuter tp1.exe en pas-à-pas**

**Sous-programme**

* Noter la valeur du registre SP ***FFF2*** (SP = Stack Pointer = pointeur de pile)
* dans le panneau de code, noter l'adresse de l'instruction qui suit immédiatement le call \_foncAvecAssembleur (= adresse de retour) ***02FA***

touche F7

* A quel endroit du programme se trouve-t-on maintenant ? ***au début de \_foncAvecAssembleur***
* Quelle valeur y a-t-il dans le registre SP ? ***FFF0***
* Dans quel sens SP a-t-il évolué ? ***vers les adresses basses (il a diminué)***
* Dans le panneau de pile (en bas à droite) observer l'adresse correspondant à la position du triangle,
* que repère ce triangle ? ***il repère l'adresse présente dans SP***
* Le registre SP repère le sommet d'un endroit de la mémoire appelé la pile,
* quelle est la valeur stockée en mémoire au sommet de la pile ? ***02FA (l'adresse de retour notée précédemment)***
* De manière générale, observer l'incrément entre deux adresses consécutives sur la pile. ***il est de 2***
* Quelle est la taille des mots visualisés sur la pile ? ***des mots de 2 octets***

**Constante**

* Exécuter avec F8 les trois premières instructions,
* noter la traduction de CONSTANTE en sa valeur 5.
* touche F8
* vérifier la valeur du registre AX. ***0005 ce qui est bien la valeur de CONSTANTE***

**Variable globale count**

* on est maintenant sur le point d'exécuter mov [\_count],ax,
* noter la présence des crochets autour de \_count (optionnels pour le programmeur mais Turbo-Debugger les met)
* observer dans le code machine l'adresse de la variable count (dont les deux octets sont inversés) ***adresse 037E***
* aller dans le panneau de données et se positionner sur cette adresse ***037E*** avec Ctrl+G,
* exécuter un nouveau pas de programme avec F8,
* observer l'écriture dans la variable globale count,
* vérifier que la bonne valeur a été écrite. ***on voit que la valeur 0005 est écrite, comme la mémoire est visualisée octet par octet, le poids faible et le poids fort sont inversés***

**Paramètre**

* Passer le printf avec plusieurs F8 et s'arrêter sur mov ax,[\_count],
* que vaut ax ? ***001C*** Quelle instruction a bien pu modifier sa valeur ? ***une instruction faisant partie du sous-programme \_printf***
* touche F8,
* vérifier que ax reprend la valeur de count, ***ax reprend la valeur 0005***
* observer la correspondance que le compilateur a établi entre param et [bp+4],
* calculer la somme "valeur du registre BP" + 4 ***BP=FFEE, FFEE+4 = FFF2***
* il s'agit d'une adresse sur la pile,
* observer cette adresse sur la pile (ce n'est pas le sommet),
* noter la valeur présente dans la pile à cette adresse, ***0005***
* touche F8,
* vérifier la valeur de ax (résultat de l'addition de count et param), ***ax prend la valeur 000A (count=0005 et param=0005, 0005+0005=000A)***

**Variable locale**

* touche F8,
* comment le compilateur a-t-il transcrit l'instruction asm mov [result],ax ? ***mov [bp-2],ax***
* selon le même principe que précédemment, prévoir l'action de l'instruction mov [BP-2],ax sur la pile ***BP vaut FFEE, BP-2 vaut donc FFEC, la valeur contenue dans AX (000A) est donc écrite dans le segment de pile à l'adresse FFEC***

**Résultat d'une fonction C**

* Observer comment le compilateur a transcrit le return result; en assembleur, ***mov ax,[bp-2]***
* noter le nom du registre que le compilateur utilise pour retourner le résultat de fonction C, ***il s'agit du registre AX***
* c'est une convention à connaître.

**Fin de sous-programme**

* appuyer plusieurs fois sur F8 et s'arrêter sur l'instruction ret,
* noter la position du sommet de pile ***FFF0*** et la valeur qui y est stockée, ***02FA***

touche F8,

* noter la valeur de IP ***02FA*** et la position que cela représente dans le programme, ***pop cx juste après le call \_foncAvecAssembleur***
* vérifier qu'il s'agit bien de l'adresse de retour notée précédemment ***nous avions noté 02FA***
* terminer l'exécution du programme rapidement avec F9.
* Consulter l'écran utilisateur (menu Window / User screen ou Alt+F5)

***on peut aussi utiliser du C***
***count=5***
***res=10***

* vérifier la cohérence des résultats affichés. ***La fonction foncAvecAssembleur( 5 ) écrit dans count la valeur de CONSTANTE soit 5, et retourne comme résultat count + param = 5 + 5=10, param prenant la valeur 5 à cause de la valeur 5 passée en argument : foncAvecAssembleur( 5 ) le résultat de la fonction est ensuite mis dans res : res=foncAvecAssembleur( 5 )***